

Pontiac G8 “Atari Gauge” Cluster Reprogramming to 3-Gauge Display.

The 2008 Pontiac G8 has a 2-gauge display above the radio nicknamed the “Atari Gauge” for its lackluster design. By default, it displays oil pressure and battery voltage in a very 1980s appearance. Some Holdens had an upgraded display that displayed battery voltage, oil pressure, and oil temperature, but in a much more sporty design. The factory gauge can be reflashed to display these. The gauge came factory on 2008s, but not 2009s. The gauge also works on 2009 models, as long as you have the wiring harness with the gauge. The plug for them is capped off on 2009s. The plug is to the left of the radio when you remove the kick panel near the gas pedal. Gauge part numbers: 92200358, 92217146, 92228638, or 92219491. Either can be used. Harness part number: 92182437.



DISCLAIMER: I am not responsible for any damage caused by modifications you choose to make. This will involve soldering to the main board of the gauge and flashing firmware files. There is potential to permanently damage the gauge, rendering it unusable. There are users and companies that provide this service for a high price. For the true DIY modder with some electrical experience, this is how it is done. Proceed at your own risk.

For those who have already used my earlier radio programming DIY guide, you will already have one of the tools and knowledge required for part of the programming. I will not go into depth on connecting the EEPROM programmer and flashing, as that is thoroughly detailed in the previous guide. You can view that guide here: <https://drive.google.com/file/d/0BwykeXNkauG0X3lWVlB5bz1lS2s/view>.

There are two chips to program: an EEPROM, and an MCU. The MCU is programmed using Serial/TTL interface. Programming the EEPROM will use a standard clip-on to USB programmer like the radio guide, but programming the MCU is a bit more involved. The MCU is an M16C/M32C family chip. I suggest reading the chip's manual to get an idea of how the structure works. Particularly, the programming instructions come from Chapter 21: Flash Memory Version, under Standard Serial Programming. <https://drive.google.com/open?id=1tuZVtjw0BPLxxHSq46Itrq1oL-D0hj9b>. It's an extensive read, but it will help you understand basic MCU processes.

TOOLS REQUIRED:

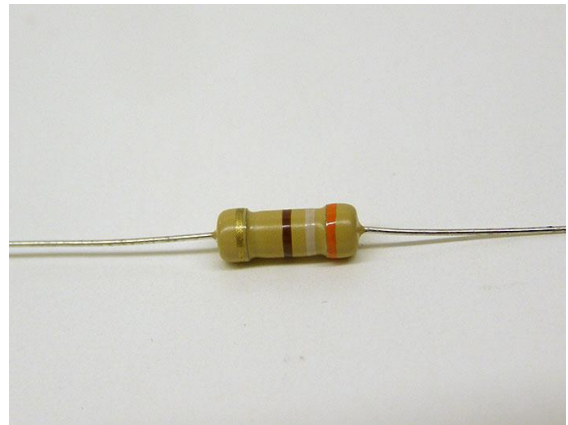
24Cxx clip/wire
EEPROM USB programmer
5V Serial to TTL Converter cable (or USB to TTL cable)
Soldering Iron
4.7K ohm resistor
12V DC power source with ground

SERIAL/USB CABLE

The MCU is programmed via a standard serial port connection. This can be accomplished on a PC using a standard serial port, or a USB/Serial adapter. I use a USB/Serial adapter due to the lack of a serial port on my PC. True serial connection is more reliable, with less interference from Windows Drivers. Add-on Serial cards are able to be installed in a PC, but with no other use for serial ports, I opted for USB adapter. Either method, the connections will remain the same. It must have 5V output.

There are MANY TTL cables out there to choose from. The cable I used is a USB TTL Serial cable from FTDI, the **TTL-232RG-VSW5V-WE**. Found here for \$22:

https://shop.clickandbuild.com/cnb/shop/ftdichip?productID=149&op=catalogue-product_info-null&prodCategoryID=296.



For this FTDI cable, you must install current Virtual Comm Port drivers found here:

<http://www.ftdichip.com/Drivers/VCP.htm>. Plug the cable in, and install the drivers. This will vary if you use other cables, or a true Serial method. Virtual Comm Port drivers will be needed for a USB connection.

Download and install M16C Flasher Tool: <https://drive.google.com/open?id=1q2BjHroK-JDxVmVWJdFJh7cftCqXEpln>. This file is required for any connection method (USB or Serial)

When accessing the chip, there are 5 “channels” to connect: **Tx**, **Rx**, **Vcc**, **Gnd**, and **Clk**. Whether USB or Serial is used, these are the connections needed.

Tx = Transmit (Sends data)

Rx = Receive (Receives data)

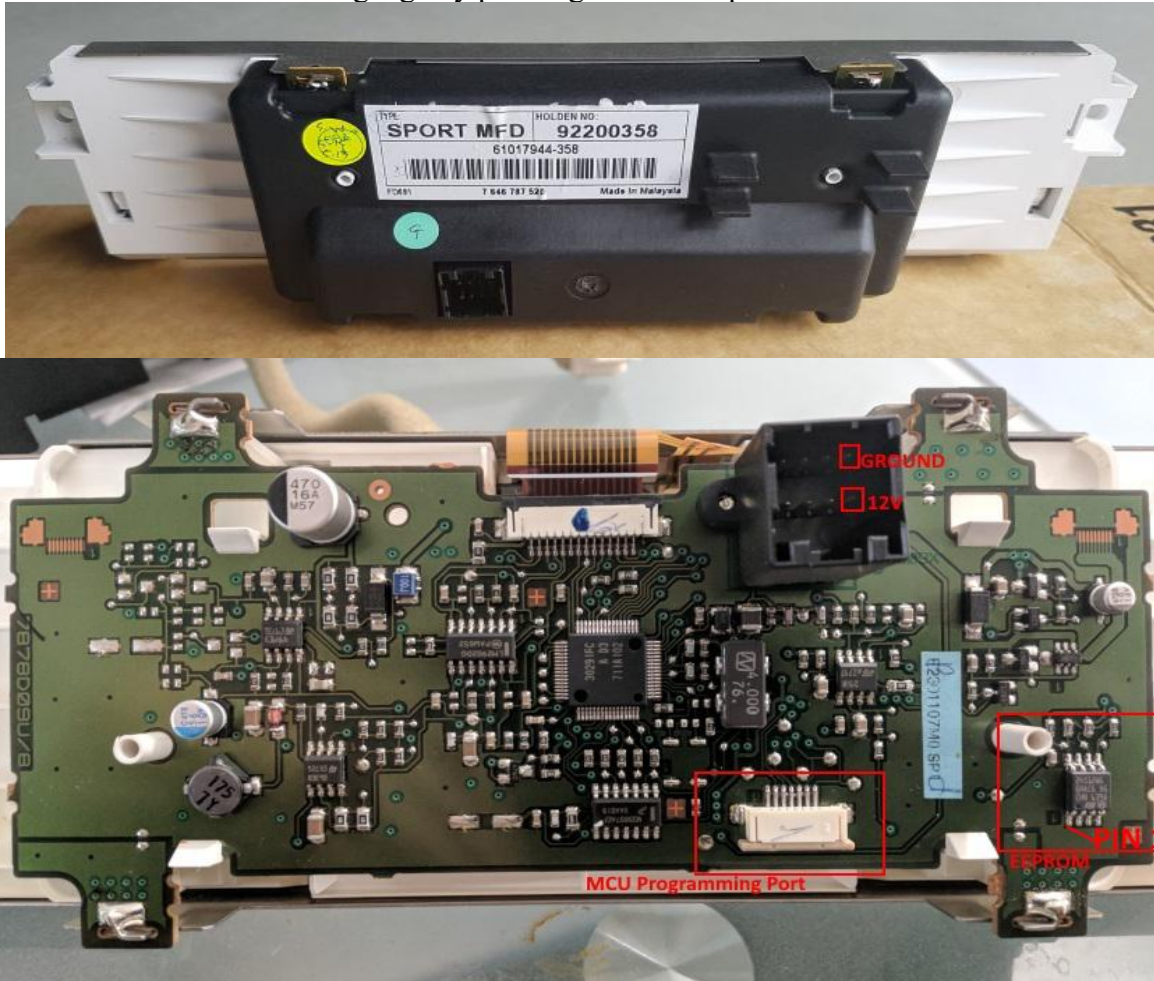
Vcc = Voltage (Provides power, in this case, 5V)

Gnd = Ground

Clk = Clock (Also seen as SCLK for Serial Clock or Clock Signal, it tells it what MODE to be in)

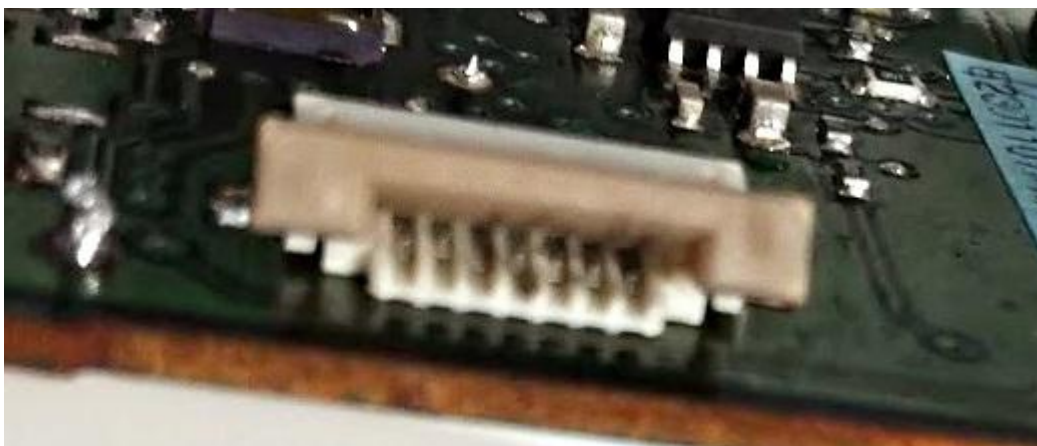
SETUP

Remove the black cover from the gauge by pressing in the 2 clips on the sides.



When connecting the wires to the chip or connecting power to the gauge, do not have the cable connected to the PC yet. Connect all programming wires before connecting to the PC.

Lift up the latch on the programming port, and there are 7 pins. From left to right, numbered 1-7, we are only using pins 1, 3, 4, 5, and 6.



PIN LAYOUT USED FOR PROGRAMMING

1: Gnd

3: Vcc

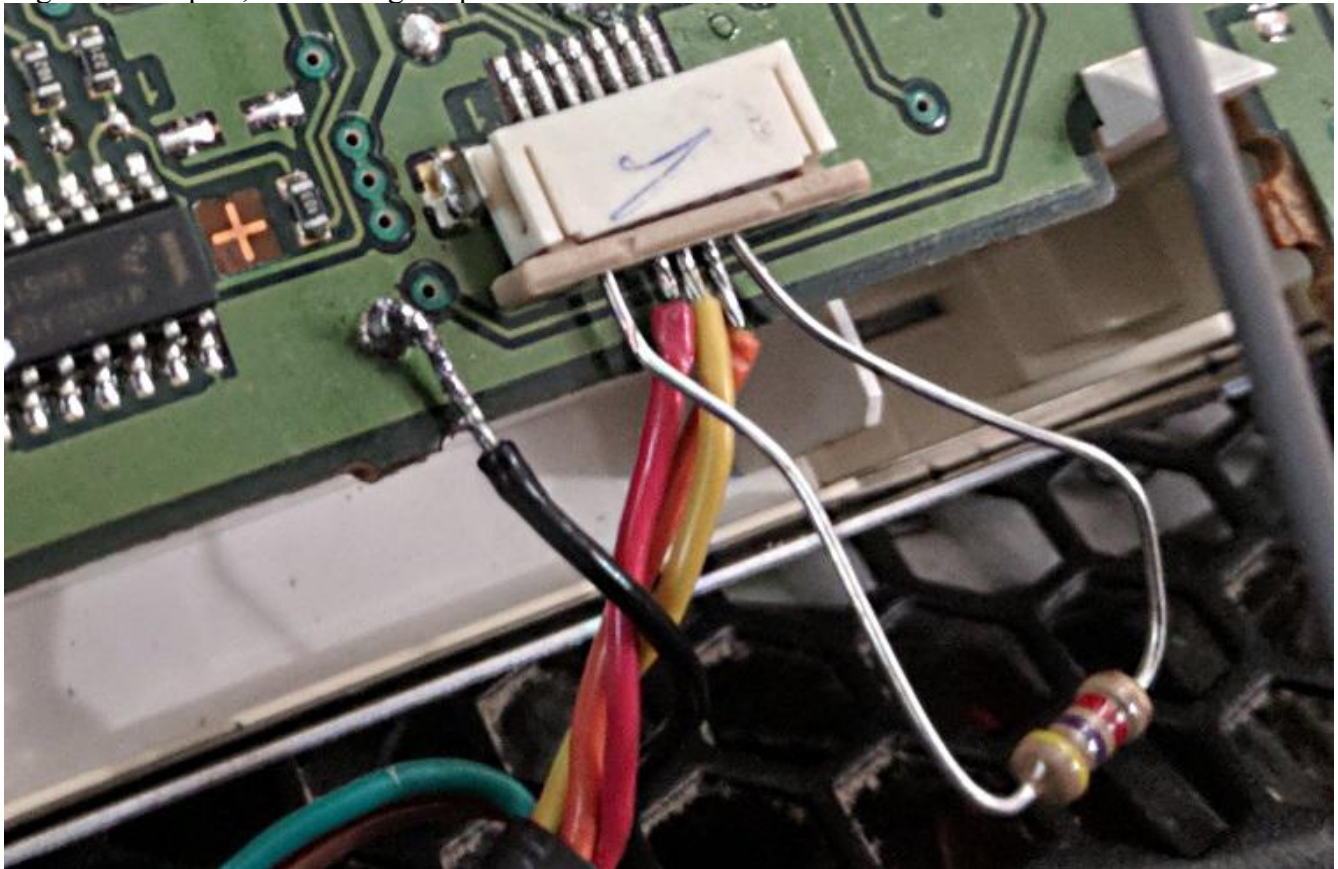
4: Tx

5: Rx

6: CLK

Pins 2 and 7 are not used. During serial programming, the cable's Tx is connected to the chip's Rx, and the cable's Rx is connected to the chip's Tx (One transmits, one receives & visa versa). On the FTDI cable I used, red is Vcc, black is Gnd, yellow is Rx, orange is Tx, and green and brown are unused.

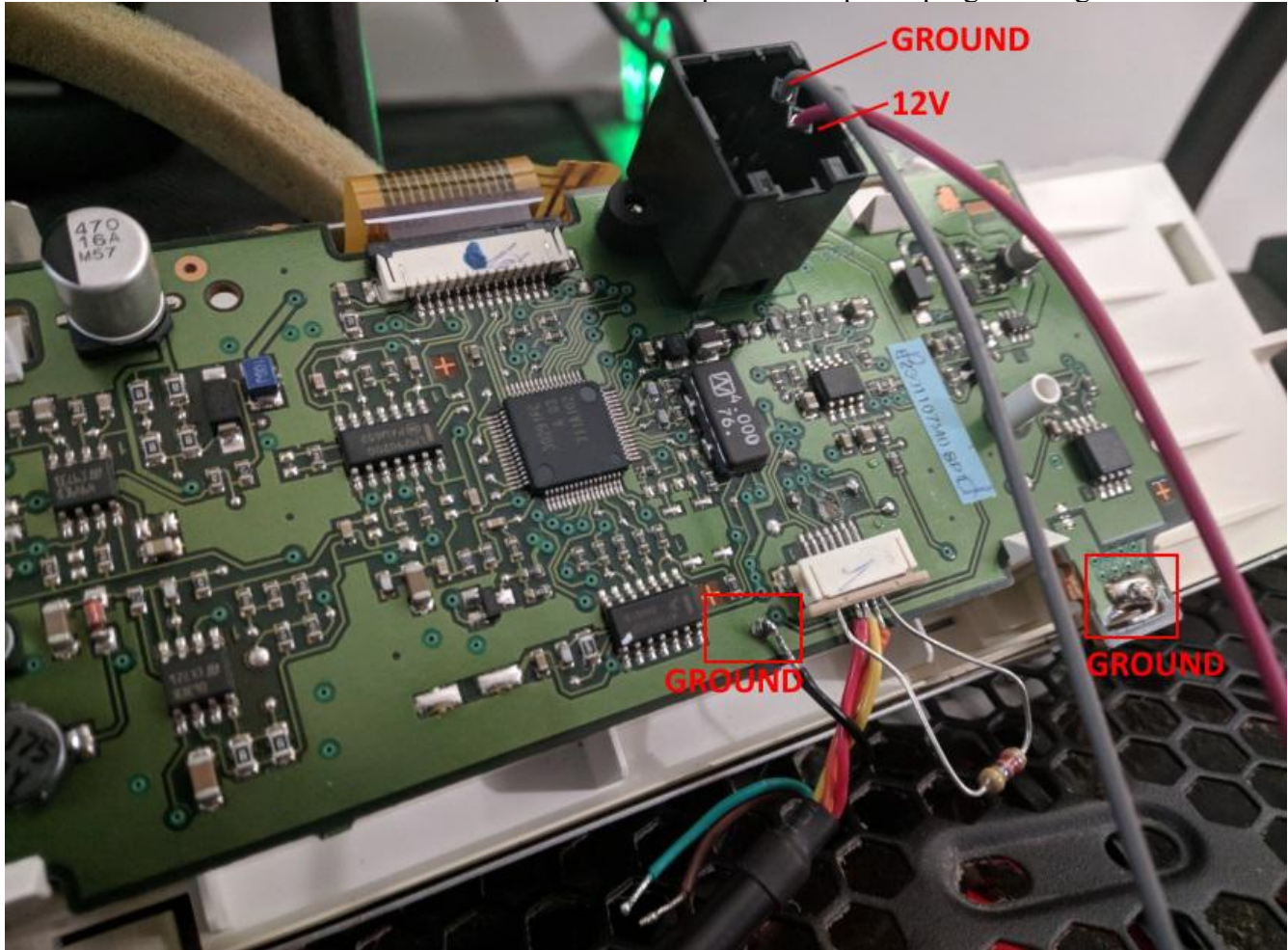
This method is done with minimal soldering, by just sliding the wire ends into the programming port to align with the pins, and closing the port lock on them to secure it.



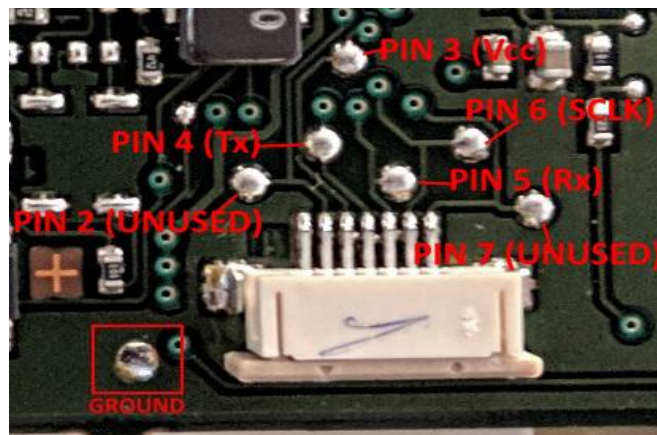
Slide the red, yellow, and orange wires into pins 3, 4, and 5 on the programming port (Or if using a different type of cable, make sure your Vcc is in Pin 3, Rx is in Pin 4, and Tx in Pin 5). Bend the 4.7K ohm resistor and slide the ends into pins 1 and 6. The Gnd wire from the cable can be soldered into any ground, I used the solder pad to the lower left of the programming port. Move the other 2 wires from the cable (green and brown) out of the way, make sure the ends don't contact anything or each other.

Once the programming port is wired in, you will need to supply an external 12V DC power source and ground to the gauge display board. I used an old PC power supply. I tapped a wire into the standard yellow 12V and black ground wires from the PC power supply, and wired into the gauge board. You can use a standard wall DC adapter, as long as it outputs 12V. I tried an old 12V adapter laying around, but it actually output 17V, and it didn't work. Test with multi-meter to be sure!

Here you can see the final setup, everything wired in, ready for programming. The power and ground are inserted into the board's main harness connector. The red wire is connected to 12V, the gray wire is connected to Ground. The resistor is required because it puts the chip into programming mode.

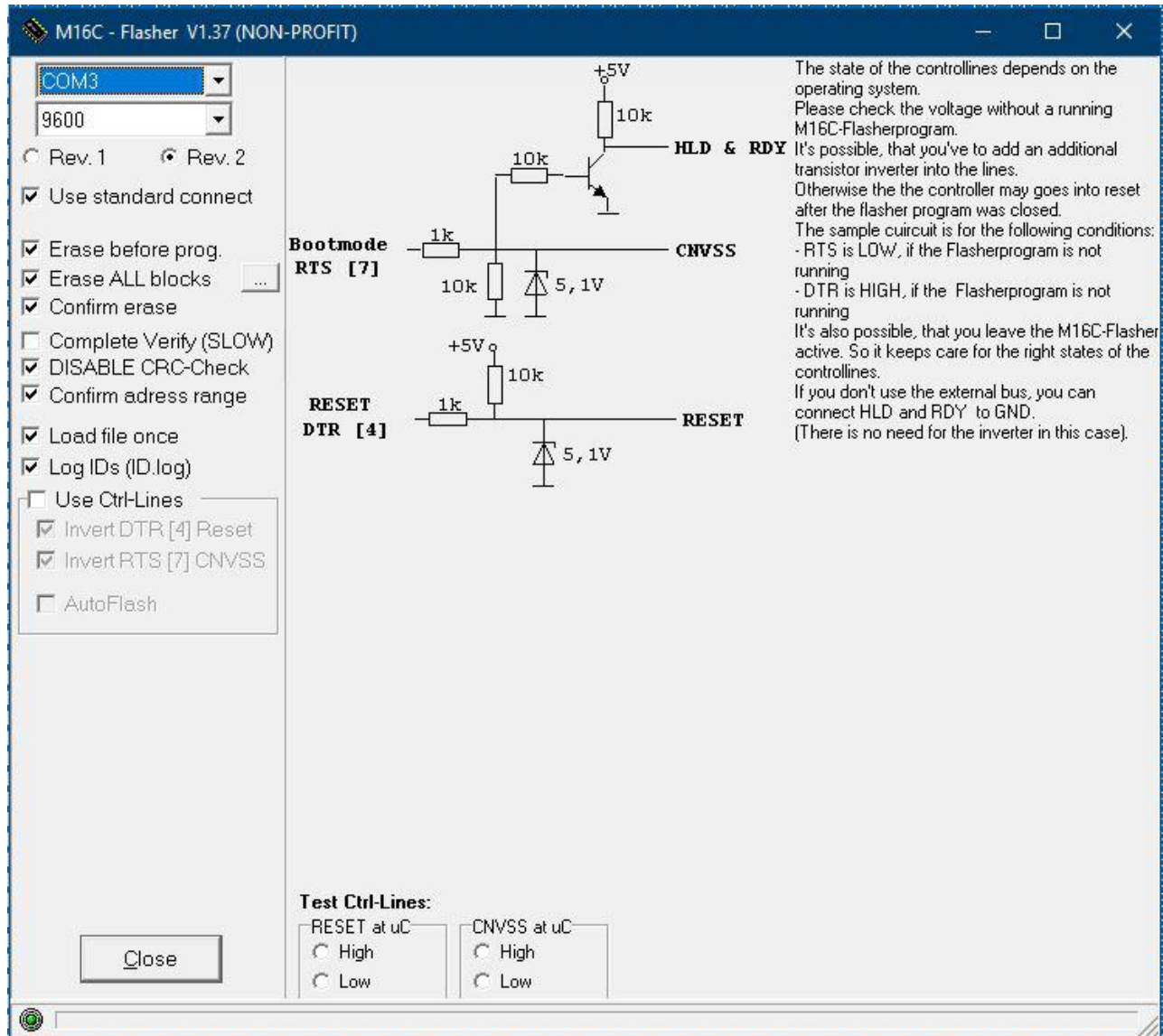


Use a digital multi-meter to ensure your connections all flow. With this connection, you can test for continuity to the board by testing the exposed wire on the cable, and tracing the soldered points on the other side of the port. Each one has a small solder pad you can trace to to test. Test the programming wires and grounds for continuity. Test 12V power for 12V. Test resistance between the resistor, and ensure it's in the range of 4.7K ohms. It may not be exactly 4.7K, but close, and it will work. Test the ground side of the resistor to ground on the board.



PROGRAMMING

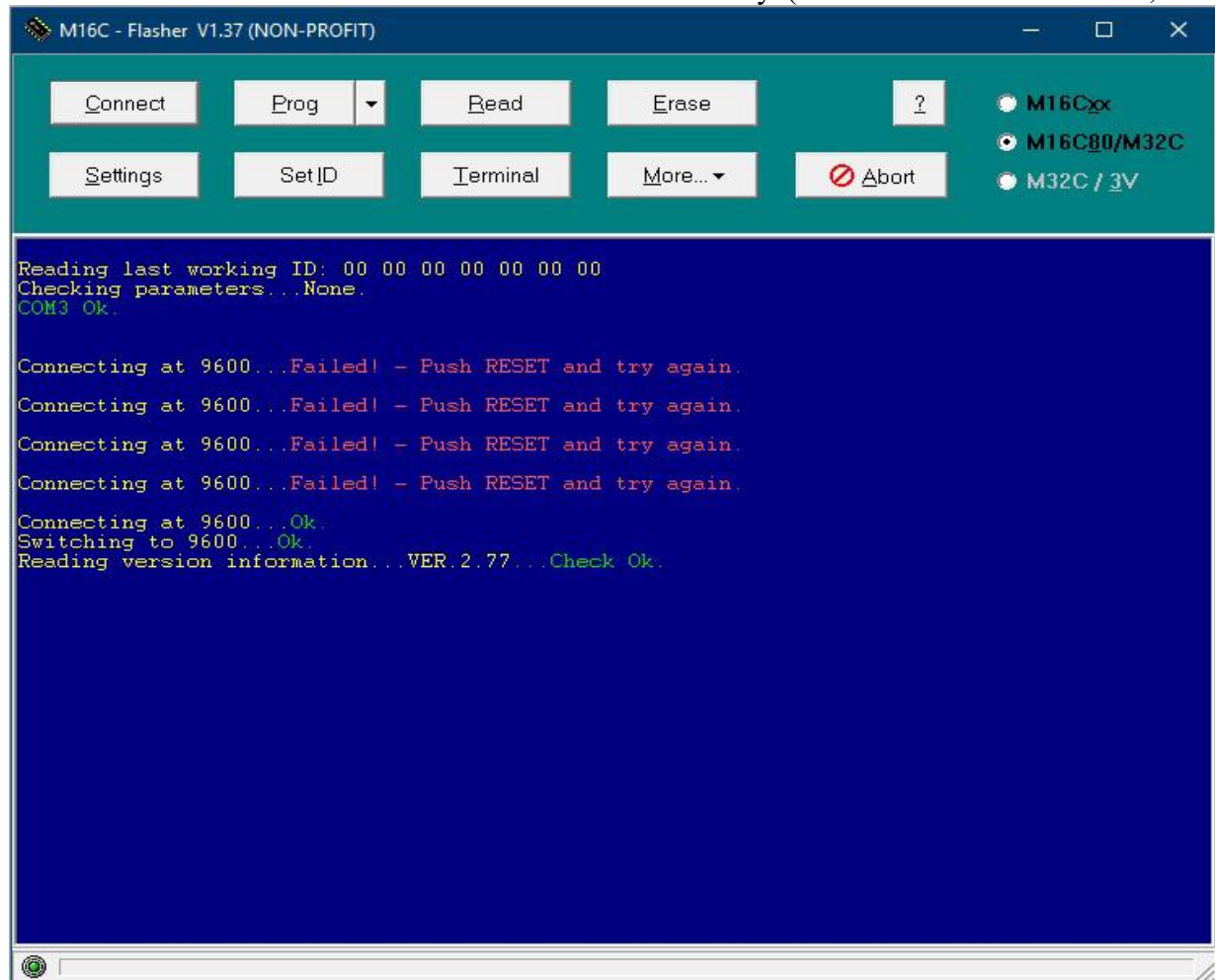
First connect the programmer to your PC, and then apply power source to the gauge. Errors will occur if power is applied before the programmer is connected to the PC. After the cable is fully wired up to the board, and the board is powered with 12V/ground, connect to PC. Check Device Manager under >Ports (COM & LPT) and find the new COM port assigned to the cable. This part may vary widely depending on your application due to different types of USB cables and/or if you use a Serial connection, but your PC will assign a COM port. Mine defaults to COM3. Open the M16C Flasher tool installed earlier, and click on SETTINGS. You will need to change the COM port from the first drop down menu to the one assigned to your cable/device. Make sure all the rest of the settings below match, and then click Close.



Back on the main screen, you should see "COM# Ok." Where # is your assigned COM port. At the top right, there are 3 settings: M16Cxx, M16C80/M32C, and M32C/3V. Make sure **M16Cxx** is checked. It is checked by default. Click CONNECT. If the setup and settings are correct, you will see "Connecting at 9600... OK, Switching to 9600...Ok, and Reading version information... VER 2.77...Check Ok."

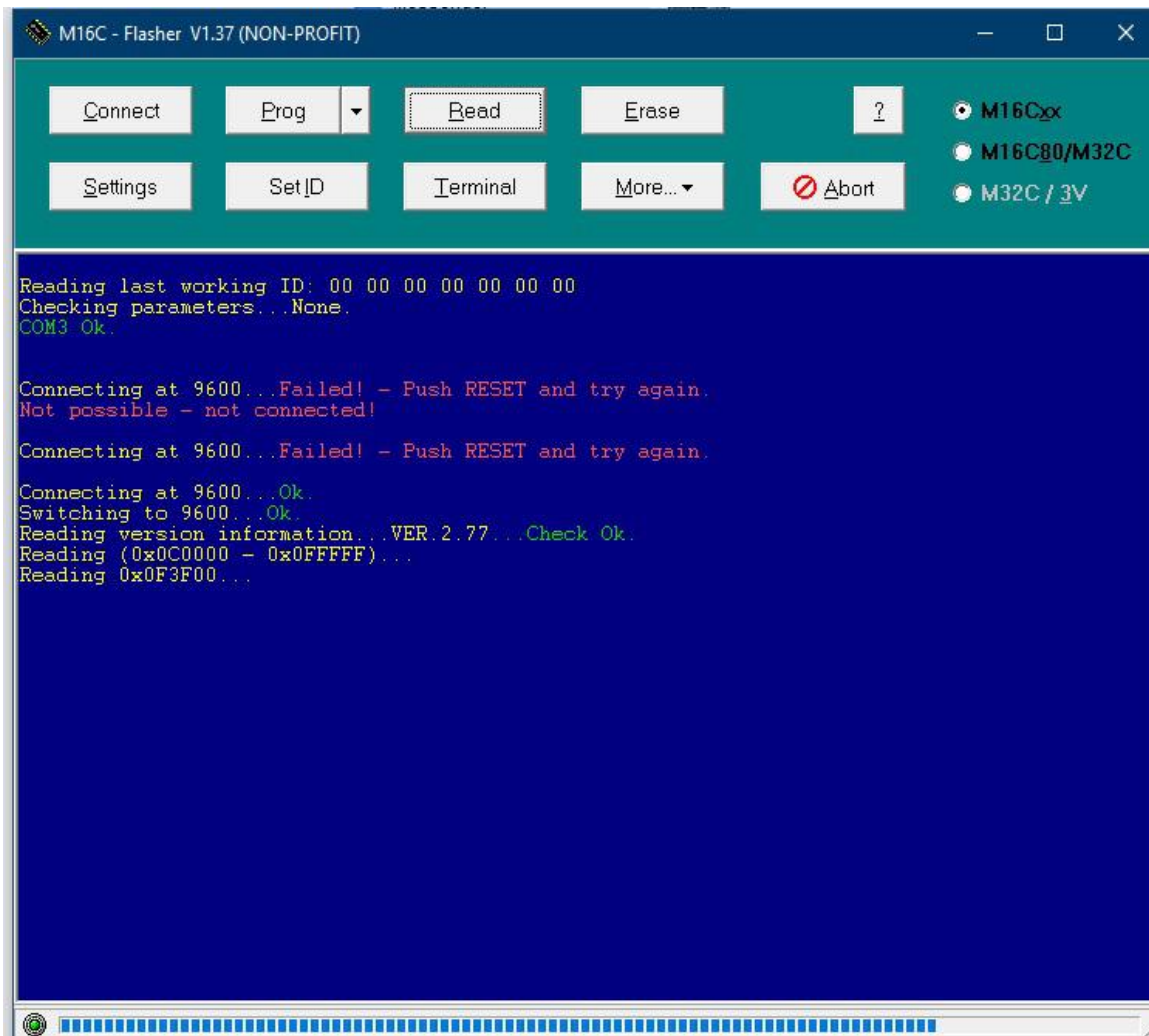
If you receive errors “Failed! - Push RESET and try again.”, then either the device is not wired correctly, the cable is not recognized by your computer, or your program settings are incorrect. If you receive a Failed message, disconnect 12V power and ground first, then diagnose your wiring and cable drivers on your PC. Most of the time I found this error, the 12V and ground connections were weak or not secure.

On this screen, it took me 5 attempts to get it to connect. Each time, I checked my wiring and 12V & grounds were tight and secure. Eventually, it connected, and later I changed my 12V and ground power supply cables, and it will connect first try, every time now. When it shows “Reading version information...VER.2.77...Check Ok” it is connected successfully. (M16Cxx should be selected, though)



If connection is successful, the FIRST step is to backup your stock 2-gauge firmware file in case of failure. Select **M16Cxx**, Click READ. It will ask for an address range, default is 0x0C0000 – 0x0FFFFFF, click OK, and then it will ask you to save the file. Name it, and save it on your PC. It will take several minutes. During the backup, the screen will display Reading “(0x0C0000 – 0x0FFFFFF)...” and a blue progress bar will display on the bottom.

If an ID: mismatch error occurs during Read attempts, try updating your Virtual Comm Port drivers (worked for me). If using the FTDI cable, they're available on the FTDI website for the cable. For an MCU, an ID code is used as a locking password to prevent programming. The factory left this chip unlocked, so the default ID is 00 00 00 00 00 00 00. An ID can be programmed into it, for example an aftermarket company who programs MCUs could do it to prevent copying of their software.



BACKUP/READ COMPLETE

Download the 3-gauge MCU firmware here: https://drive.google.com/open?id=1q95WIRiD6fUK_EYSoGL_I-7OoMo0dYiO

Once the backup is complete. Ensure the mode at the top right is **M16Cxx**. Click “PROG” > “Prog new File”. Select the “G8 3 gauge SIC MCU Content.mot” file. It will ask for address range. Default is 0x0E0000 to 0x0FFFFFF. Click OK, and it will erase the chip, and begin programming.

```
Reading G8 3 gauge SIC MCU Content.mot...Ok. (0x0E0000 - 0x0FFFFFF) M16C62
Erasing selected blocks... 0 1 2 3 4 Ok.
File: G8 3 gauge SIC MCU Content.mot (6/16/2017 5:39:18 PM)
Programming (0x0E0000 - 0x0FFFFFF)...
```

Once programming is complete, it will end with an “...Ok” if successful. Remove 12V power supply & ground, disconnect the USB cable, desolder the ground, and remove all wires from the board.

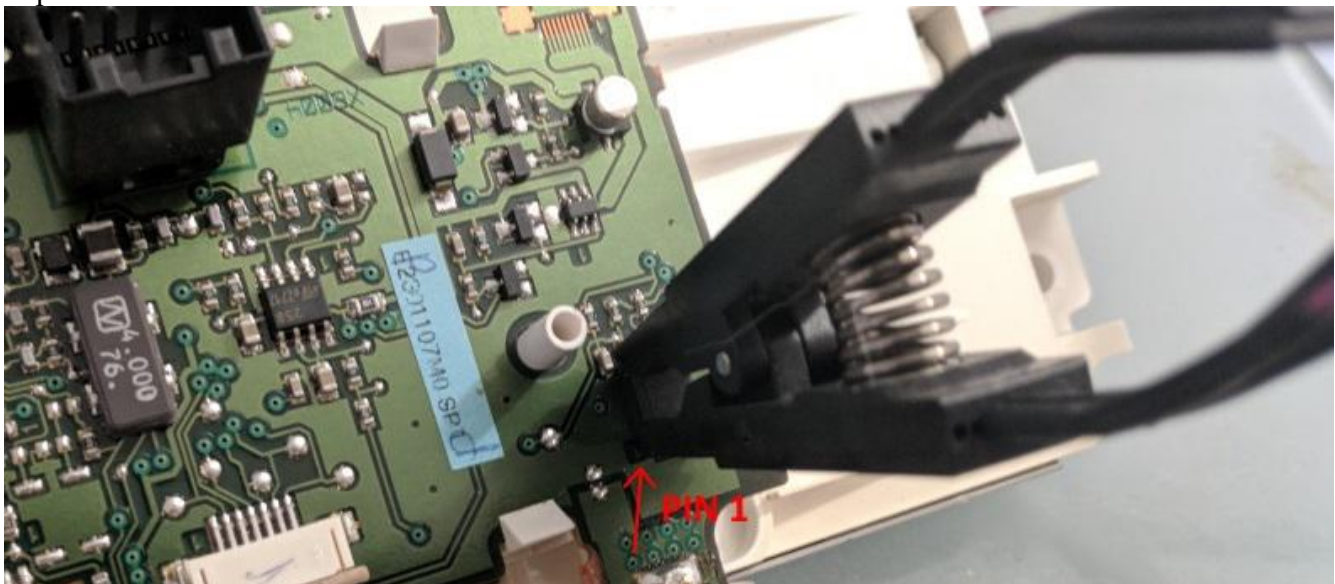
If programming fails, try again. It will have already erased your stock firmware, so you can only keep trying to program the 3-gauge file, or reprogram your stock 2-gauge file back. If you have to program your stock 2-gauge file back, the programming address must be 0x0C0000-0x0FFFFFF.

EEPROM PROGRAMMING

Finally, the EEPROM chip must be programmed for the 3-gauge file, and your VIN programmed. (Side note: I've been running my 3-gauge file with an incorrect VIN on my car since Summer 2017, and I've seen no odd occurrences or problems, so it is my understanding that these can be swapped between cars without REQUIRING the VIN to be reprogrammed.) This step is required anyway, so you should put your VIN in it.

In my radio programming guide, I went into detail on EEPROM programming including hardware and software setup, so I won't repeat it here. If you are not familiar with it, the guide is located here: <https://drive.google.com/open?id=0BwykeXNkauG0X3IWVIB5bz11S2s>.

Using your EEPROM clip, PIN1 is located at the lower left, that's where PIN1 (red wire) from your clip should attach.



Make sure to BACKUP your stock 2-gauge EEPROM .bin file immediately. In the EEPROM programming software, read the chip as a **24C512 5V** chip type. Save the backup file on your PC.

Download the 3-gauge EEPROM .bin file here: https://drive.google.com/open?id=1vUm68eL3MG2bTJH3xz_4_-WnMsbNSwml.

Open the new 3-gauge .bin file. The VIN is located at addresses 216E – 217E. In this .bin file, I've changed all the numbers to X's. On the right side of the screen with the X's, begin typing in your VIN in this location, overwriting the X's. Nothing else needs to be changed. Save this file on your PC as a backup. Click on Program.

00002160	24	32	21	00	21	32	21	32	01	32	01	32	20	32	58	58	\$2! .!2!2.2.2 2XX
00002170	58	58	58	58	58	58	58	58	58	58	58	58	58	58	58	58	XXXXXXXXXXXXXXXXXB
00002180	4F	53	43	48	30	34	30	30	53	4E	31	32	33	34	35	36	0SCH0400SN123456

ATARI GAUGE FLASH COMPLETE! You can now install it. It will now show Battery Voltage, Oil Temperature, and Oil Pressure, and will change between Imperial and Metric with your cluster setting.

Finally, you make any modifications to your car at your own risk. I will not be responsible for damages you cause, so please research beyond this document prior to doing the work. The nature of soldering and programming chips can potentially permanently damage your device. These are the steps to risk doing it yourself. If you are unsure, pay a professional to do it, or contact one of the many vendors online that offer this programming service. My goal is to share the information with the G8 community and not profit from it.

Stay modding!

Written by Tim Rose
Feb 2018

Forums: HoLLo
Email: 95birdman@gmail.com
IG: @pont14c

